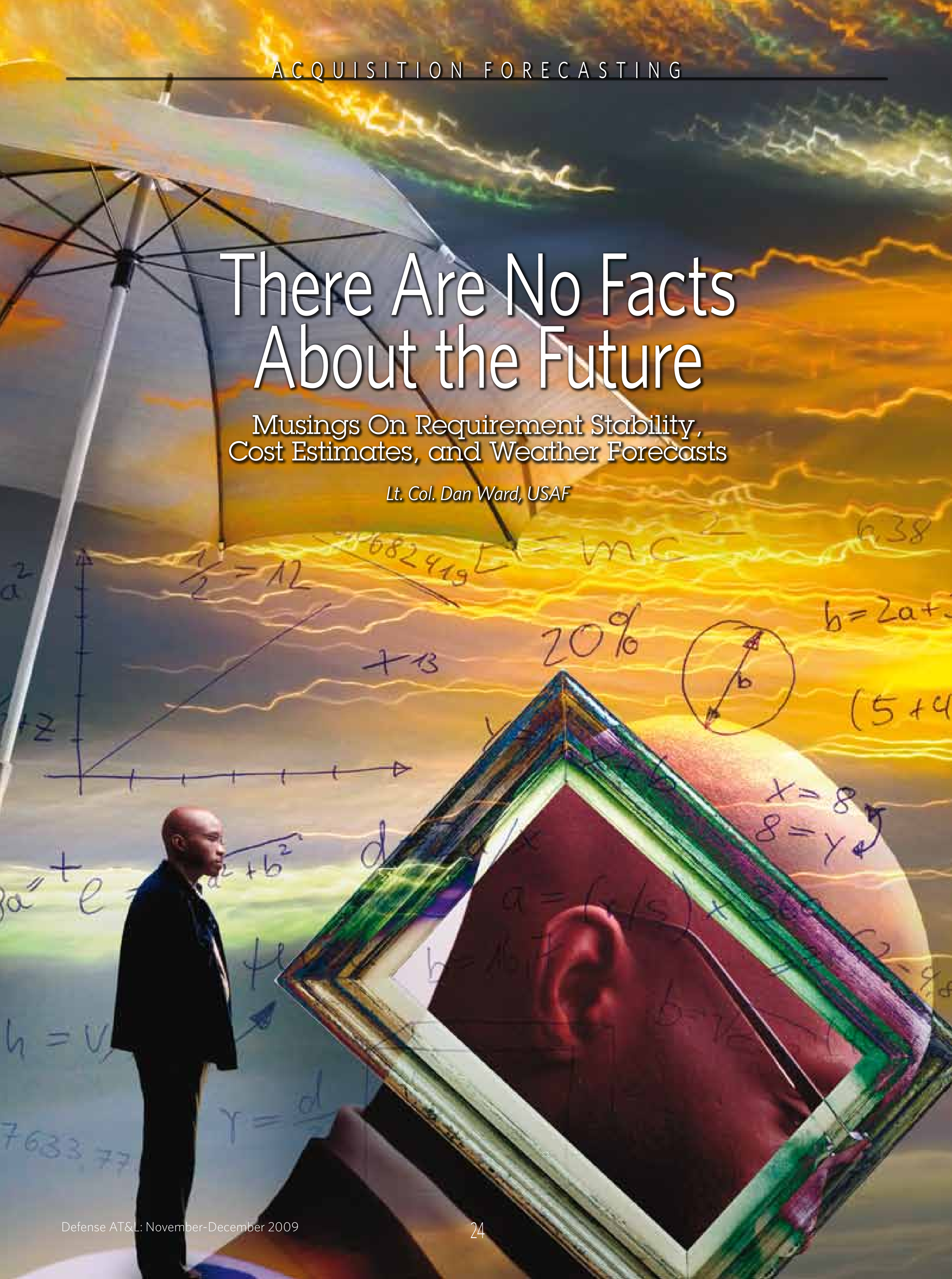


There Are No Facts About the Future

Musings On Requirement Stability, Cost Estimates, and Weather Forecasts

Lt. Col. Dan Ward, USAF

Suppose two meteorologists make predictions about the weather for a particular Saturday. Mr. Gray says there is a 50 percent chance of rain, and Mr. Blue says there is a 25 percent chance of rain. On the Saturday in question, it does indeed rain.

Which one was right: the one who predicted a 50 percent chance of rain or the one who predicted half that percentage? Were they both right? Was Gray twice as right as Blue? Can we say that one prediction was more reliable, more useful, or more accurate than the other? I don't think we can.

Yes, But Should We Take the Umbrella?

Have you ever stopped to wonder what it really means when we say there is a 50 percent chance of rain? Does it mean that on 100 Saturdays with these initial conditions, 50 will get rained on? Or is the forecaster simply 50 percent sure it will rain this Saturday, whatever that means? Is there a difference? And does either interpretation of the prediction affect our behavior? Should we take half an umbrella when there is a 50 percent chance of rain? Do we develop half a backup plan? Or simply go to the museum instead of having a picnic on half such days? And what if we pick the wrong half?

More to the point, if a 25 percent prediction of rain and a 50 percent prediction of rain can both point to a rain event and say, "See, I told you it might rain!" (or point to a non-rainy day and say much the same thing), what value is there in the prediction? How does this prediction help with our planning or execution?

There Are No Facts About the Future

Jeff Wacker, a fellow at the EDS Corporation, once explained to me: "There are no facts about the future." I thought that was an interesting observation, particularly coming from someone whose job title is "futurist." It also struck me as particularly insightful and—most important—completely true. It also struck me that being a futurist could be either really fun and easy, or really frustrating and hard. It's probably both.

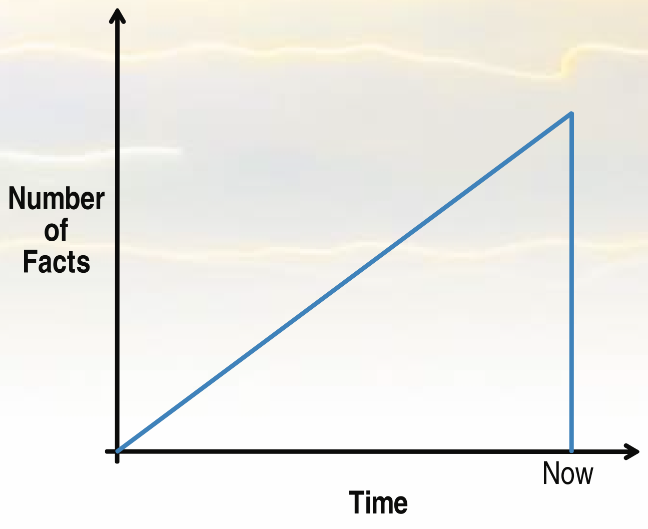
The Zero Future Facts Principle is illustrated in Figure 1. For simplicity, the number of facts in the universe is represented as increasing linearly, but this need not be the case and is not germane to this discussion. The only important fact is that beyond the Now Point, there are no facts. In the future, there are only conjectures, estimates, guesses, and predictions. We can assign a percentage to our prediction (a 50 percent chance of rain, for example), but that does not make it a fact. The only facts we'll ever encounter are about what has happened or what is happening. There are no facts about what will happen.

Ward *is the chief of process improvement and reengineering in the Acquisition Chief Process Office, Office of the Deputy Assistant Secretary of the Air Force for Acquisition Integration. He holds degrees in systems engineering, electrical engineering, and engineering management. He is Level III certified in SPRDE and Level I in PM, T&E, and IT.*

This principle has serious implications for program managers, as research by the Standish Group demonstrates. Performing research on project success and failure since 1985, Standish Group reports bluntly state that estimates come in two categories: lucky or lousy. According to their research, "there is no such thing as a reliable estimate. Learning to work better with poor estimates rather than developing better estimating techniques is crucial." One more time: there are no facts about the future.

Yet we make programmatic estimates and predictions all the time and somehow end up treating these things as facts. As Dr. Roger Atkinson poignantly observed, our projections about time and costs are "at best only guesses, calculated at a time when least is known about the project." We should be mindful of this when looking at cost estimates for a 10-year project or statements of operational requirements for the year 2020.

Figure 1. **The Zero Future Facts Principle**



Reflective Practice and Practical Reality

In order to help project leaders deal with these future non-facts, I have assembled a handful of charts and equations that are presented here for your consideration. Unlike dubious weather forecasts and cost estimates, the figures and formulas that follow are emphatically *not* based on quantitative research data. Instead, they are the result of a "reflective practice" methodology, as described by Donald Schön's book *The Reflective Practitioner*.

The discipline of reflective practice has a much stronger basis in practical reality than the so-called scientific attempts to predict with probabilities, which tend to be academic and idealized. In contrast, reflective practice primarily relies on experienced intuition and tacit knowledge—what the late Air Force Col. John Boyd, military strategist, called "fingerspitzengefuhl." My introduction of this foreign word, combined with a reference to a dead authority figure, is done to enhance the perception of the methodology's legitimacy among those who value the trappings of scientific thought. Greek letters, Latin phrases, square brackets, mathematical terms such as "absolute value," and subscripts will be used in subsequent paragraphs for the same reason.

Since most system development efforts begin with requirements, that's where we'll start too. In many projects, a dedicated effort is made to ensure the requirements remain stable. I have even heard talk of freezing requirements at the time the contract is awarded. However, requirement stability is not always desirable or beneficial. Over time, the value and relevance of a requirement degrades, either because of advances in technology that render the required capability technically obsolete or changes in the threat environment that render the requirement operationally irrelevant. Or maybe the requirement simply wasn't very good to begin with.

Increasing the project's duration increases its exposure to any number of change events, and the impact on the project is exponentially proportional to the sum of all the delays.

Thus, I developed the Law of Requirement Stability, which states, "The viability of a stable requirement drops off at an exponential rate over time." This assumes the initial requirement was good. In the event of a poor initial requirement, the value drops off much faster.

Former Secretary of Defense Gordon England expressed his support for this law in his June 3, 2009, testimony to the House Armed Services Committee's Defense Acquisition Reform Panel. England said, "Over time, they [requirements] actually do have to change. ... It's a reality of design and production and things. You want them to change. ... It's not all bad to change requirements as a program proceeds." It appears requirements do indeed have a limited shelf life.

A corollary to this law is the Law of Stupid Stability, which states, "The stupidity of making a requirement stable is directly proportional to the timeframe over which the stability is enforced." According to the law, as a requirement resists changes over a longer period of time, the likelihood of it being stupid is increased. [Technical Note: The term "stupid" is a formal engineering term that refers to requirements pursued despite being technically obsolete, operationally irrelevant, or both]. Obviously, the timeframes in question for these two laws will vary depending on the type of technology being considered. The requirements for a piece of electronic equipment, for example, will likely become stupid at a faster rate than the requirements for a suspension bridge. It is also worth noting that these two laws apply directly to individual requirements and apply exponentially to documents containing multiple requirements.

Clearly, these laws illustrate the importance of *stable* requirements over short timelines and flexible requirements over long timelines. Some might even suggest they illustrate the importance of short timelines, *à la mode*, *a priori*, and *sine qua non* [see earlier statement about the use of foreign phrases].

More Laws and Theorems

Another way of depicting the relationship between requirement flux and time is with the Requirement Shelf-Life Ratio Theorem, which states, "The amount of time spent developing a system (T_d) should be shorter than the mean-time between legitimate requirements change (T_{rc})." Mathematically, this is represented thus: $T_d < T_{rc}$. [Note the use of subscripts, which is a universally acknowledged sign of scientificality.]

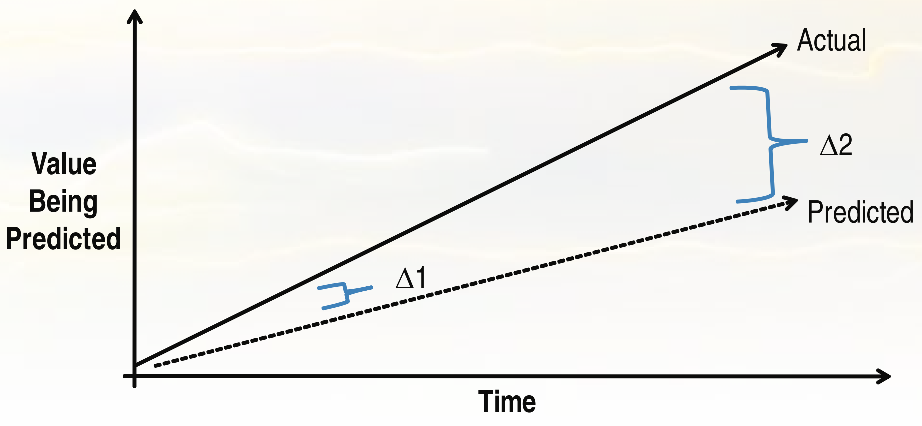
This theorem explains that if the amount of time spent developing a system exceeds the sum of the requirements' shelf-life, the resulting system will be operationally irrelevant and/or technically obsolete (i.e., "stupid") when delivered. As with the earlier laws, the Requirement Shelf-Life Ratio Theorem suggests that short timelines are much to be desired.

Once the requirements are established and the development work begins, project leaders are often faced with opportunities to delay decisions and push significant events to the right. The previous laws and theorems notwithstanding, there is a widespread belief that schedule extensions improve the project's outcome. Our research shows that such extensions should be avoided at all costs because of the Law of Increasing Impact, which states, "The impact of a delay increases exponentially with the length of the sum of the delays."

Figure 2. **Zeno's Donut of Conundrum**

Delay
Causes
Delay
Causes

Figure 3. **The Law of Error Propagation**



Let's take a closer look at why this law is true. Over a given amount of time, projects are exposed to a certain amount of change. Over a longer timeframe, they are exposed to a greater quantity of change events. These change events increase the risk of technological obsolescence; budget instability; operational irrelevance; personnel transfer (which causes a loss of learning, accountability, and consistent leadership vision and support); and non-compliance with new regulatory requirements. Increasing the project's duration increases its exposure to any number of change events, and the impact on the project is exponentially proportional to the sum of all the delays.

Close study of the Law of Increasing Impact intuitively reveals the Recursive Delay Self-Propagation Theorem, which states, "The length of a delay increases with the length of the delay." That is, any increase in a project's development schedule will cause an additional increase to the project's development schedule. Mathematically, this can be expressed as

$$D = D + \sin\alpha\sin\beta + 2\sin\tfrac{1}{2}(\alpha+\beta)\cos\tfrac{1}{2}(\alpha-\beta) + a_0 + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{L} + b_n\sin\frac{n\pi x}{L}\right)$$

or more simply, as D = D + n, where D is the delay and n is some number. The proof is left as an exercise for the reader.

A concrete example may shed further light on this principle. Consider that a sufficiently long delay in a project leads to instability among project personnel (resulting from retirements, promotions, transfers, etc.), which leads to additional delays as the new personnel are hired, trained, and brought up to speed on the project. Each new person introduces additional change, which exacerbates the whole situation and causes more delay, making progress impossible. This principle is also known as Zeno's Donut of Conundrum, and is illustrated in Figure 2. [NOTE: Zeno's Donut of Conundrum is named after my uncle Zeno, and is not to be confused with Zeno's Paradox of Motion from Greek philosophy ... but now that I think about it, they're quite similar.]

No Certainty but Uncertainty

Let us now return to the central theme of uncertainty. Because there are no facts about the future, our estimates of future values are sometimes incorrect. For the sake of appearing scientific, we use the Greek letter delta (Δ) to represent the absolute value of the difference between a predicted value and the actual value. As an error propagates over time, the value of Δ increases according to the Law of Error Propagation (see Figure 3), which states, "The absolute value of the Δ between an actual value and an erroneously predicted value increases in direct proportion to the time over which the error is propagated." So given sufficient time, small errors become big errors. As we see in Figure 3, Δ2, which occurs later in time, is much larger than Δ1.

The Case for Short Timelines

Taken together, all of these laws, theorems, and principles make a strong case for using short timelines when developing a new system. This perspective is emphatically supported by countless Government Accountability Office reports, one of which explained: "A hallmark of an executable program with a sound business case is *short development cycle times*" (Report on Selected Weapon Systems, GAO, 2008, emphasis added).

In much the same spirit, The Standish Group explained in a 1999 report that "we have long been convinced that shorter time frames ... increase the success rate." In their February 2008 newsletter, The Standish Group was more emphatic, writing that "with projects, slow kills; speed increases success." A particularly fierce report by The Standish Group observes that "time is the absolute enemy of all projects." Time therefore joins Al Qaeda and the Taliban in the latest Axis of Evil.

At this point, we briefly deviate from our preferred reflective practice method and introduce some actual research data, compliments of the aforementioned Standish Group. Some readers may wish to skip this section, and we completely understand. In our own defense, we should point out that these are someone else's data, not our own original research. We did not collect it and are merely reflecting on it. Also, note that the percentages in Figure 4 are measurements from the past, not predictions about the future.

The figure, Project Duration, Size Affect Success, presents five years' worth of data gathered by The Standish Group from IT projects. It shows the correlation between project duration, team size, and project success.

THERE'S NO TIME TO WASTE

DoDTechipedia

America's warfighters need solutions in the field fast. We're helping make that happen on **DoDTechipedia**.

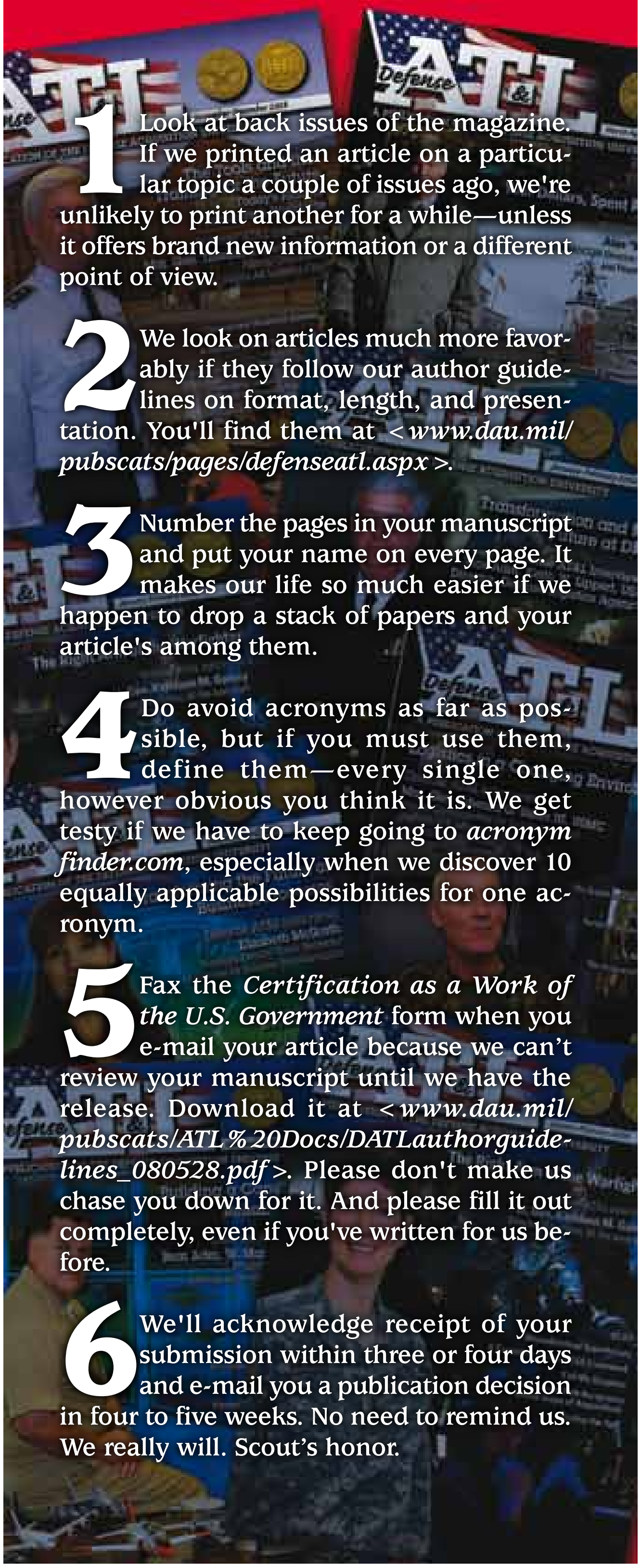
More than 9,000 defense science and technology experts are already online. Join us at **https://www.DoDTechipedia.mil**.

An internal wiki from the United States Department of Defense.

Featured in the White House Open Government Innovations Gallery.

DoDTechipedia is a project of the Under Secretary of Defense for Acquisition, Technology and Logistics; Director of Defense Research and Engineering; Defense Technical Information Center; and Assistant Secretary of Defense for Networks and Information Integration/DoD Chief Information Officer.

A Six-Pack of Tips for Defense AT&L Authors



1 Look at back issues of the magazine. If we printed an article on a particular topic a couple of issues ago, we're unlikely to print another for a while—unless it offers brand new information or a different point of view.

2 We look on articles much more favorably if they follow our author guidelines on format, length, and presentation. You'll find them at <*www.dau.mil/pubscats/pages/defenseatl.aspx*>.

3 Number the pages in your manuscript and put your name on every page. It makes our life so much easier if we happen to drop a stack of papers and your article's among them.

4 Do avoid acronyms as far as possible, but if you must use them, define them—every single one, however obvious you think it is. We get testy if we have to keep going to *acronymfinder.com*, especially when we discover 10 equally applicable possibilities for one acronym.

5 Fax the *Certification as a Work of the U.S. Government* form when you e-mail your article because we can't review your manuscript until we have the release. Download it at <*www.dau.mil/pubscats/ATL%20Docs/DATLauthorguidelines_080528.pdf*>. Please don't make us chase you down for it. And please fill it out completely, even if you've written for us before.

6 We'll acknowledge receipt of your submission within three or four days and e-mail you a publication decision in four to five weeks. No need to remind us. We really will. Scout's honor.

Figure 4. **Project Duration, Size Affect Success**

Project Size	People	Time (mos.)	Success Rate
< $750K	6	6	55%
$750 - $1.5M	12	9	33%
$1.5M - $3M	25	12	25%
$3M - $6M	40	18	15%
$6M - $10M	250+	24+	8%
Over $10M	500+	36+	0%

The results are striking, even if we chose to disbelieve the measured success rate for the largest projects. (Really? *None* of them succeeded? OK, I guess that's not too surprising.) The overall trend clearly shows that success and duration are inversely proportional, historically speaking. As reflective practitioners, what, then, should we make of this in terms of practical, actionable conclusions? Perhaps the most reasonable conclusion is that when launching a new project, we should establish the shortest possible schedule and ensure that schedule slips are treated as a measure of last resort. We should never expect schedule delays to help ensure project success.

This means project leaders should be willing to descope the project before accepting a schedule delay because it is generally better to deliver something rather than nothing, and to succeed a little rather than fail a lot. Similarly, the evil practice of taking money from a project in the current year and repaying it in a future year should be avoided at all costs. Such a tactic merely pushes the work out to a future year, which causes a ripple of increasing delays, and that's not good.

In order to help keep the timeline sufficiently short, the organization should probably use a small team and provide a small budget, per Figure 4. This does not guarantee success, but it seems to increase the odds (whatever that means). Such a restrained, disciplined approach has the desirable effect of enabling a larger number of small projects across the enterprise, each of which has a greater likelihood of success,based on historical trends (if we may be allowed to make a prediction about the future). Yes, it is harder to manage and deconflict a portfolio of many small projects, but isn't it more fun to have projects that succeed? And making life easier for management isn't really the point now, is it?

In conclusion, the verdict is in. Long development timelines are *contra bonos mores* and *cogito ergo sum*. If we want to successfully deliver our projects, we'd better move fast. *Quod erat demonstrandum*.

The author welcomes comments and questions and can be contacted at daniel.ward@pentagon.af.mil.